AFRL-AFOSR-JP-TR-2017-0026

Adaptive Problem Solving

Michael Barley
THE UNIVERSITY OF AUCKLAND

03/01/2017
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ IOA
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
07-03-2017	Final	27 May 2015 to 26 Nov 2016

4. TITLE AND SUBTITLE
Adaptive Problem Solving

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA2386-15-1-4069

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Michael Barley

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
THE UNIVERSITY OF AUCKLAND
24 PRINCES STREET
AUCKLAND, 1010 NZ

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AOARD
UNIT 45002
APO AP 96338-5002

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR IOA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-JP-TR-2017-0026

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The PIs have implemented a system, MSP, that given a specific problem, automatically generates, evaluates, and assembles different combinations of representations and heuristics to create a planner for that problem. They extended this system to include evaluating, selecting, and assembling algorithms. The next generation MSP searches for the best combination of algorithms, representations, and heuristics for the specific problem given. This transforms a general domain-independent planner into an efficient domain-dependent planner.

15. SUBJECT TERMS
Planning

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			ROBERTSON, SCOTT
Unclassified	Unclassified	Unclassified	SAR	8	**19b. TELEPHONE NUMBER** *(Include area code)* +81-042-511-7008

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Final Report for AOARD Grant FA2386-15-1-4069

"**Adaptive Problem Solving**"

25 February 2017

Name of Principal Investigators (PI): Michael W. Barley
- e-mail address : barley@cs.auckland.ac.nz
- Institution : University of Auckland
- Mailing Address :
  Computer Science Department
  University of Auckland
  Private Bag 92019
  Auckland, 1142
  New Zealand

Name of Principal Investigators (Co-PI): Patricia J. Riddle
- e-mail address : pat@cs.auckland.ac.nz
- Institution : University of Auckland
- Mailing Address :
  Computer Science Department
  University of Auckland
  Private Bag 92019
  Auckland, 1142
  New Zealand

Period of Performance: 5/27/2015 – 11/26/2016

Abstract:

We have implemented a system, MSP, that given a specific problem, automatically generates, evaluates, and assembles different combinations of representations and heuristics to create a planner for that problem. In the future, we will extend this system to include evaluating, selecting, and assembling algorithms. Our next generation MSP will search for the best combination of algorithms, representations, and heuristics for the specific problem given. This will transform a general domain-independent planner into an efficient domain-dependent planner.

Introduction:

Specific Aims of this Research:

In this research, we characterize problem solvers by the algorithms, representations, and heuristics that they use. Most planners use a single approach to solving problems, i.e., for all domains they use the same algorithm, representation, and use the same technique to generate their heuristics. The drawback of this is that there is almost never one problem solving technique, one representation, or one way to create heuristics that works well on all problems/domains. There is a tradeoff between the generality of a problem solver and its effectiveness. For difficult problems in a domain, we want the problem solver to take advantage of the nature of the problem to effectively solve these problems. These are called domain-dependent planners. However, in other domains that have a different structure, these algorithms, representations, and heuristics either are not effective or do not work at all. When you want a single planner to solve problems in a wide range of domains, a more general planner is needed. These planners cannot assume much about the nature of the problem and thus must use weaker (and consequently most expensive) techniques to solve their problems. However, because they make much weaker assumptions about the domains, they can solve problems in a much wider range of domains. These are called domain-independent planners. This research attempts to transform a domain-independent planner into a domain-dependent planner. This is accomplished by having a meta-level component that dynamically customizes the general planner to one that is

specifically designed for the specific problems/domains being solved and to the current computational environment. In other words, a system that is both general, like a domain-independent planner, and effective, like a domain-dependent planner. The meta-level component does this by discovering the structure of the problem and adapting the planner to work effectively with that problem's structure.

In our proposal, our research plan for this project divided the work into four areas: (1) problem solving architecture; (2) problem representation; (3) heuristics and control knowledge; and (4) algorithms. In the area of problem solving architecture (1), we implemented a meta-level problem solver, MSP, that searches a space of problem solver designs where MSP has operators that transform problem representations and creates combinations of heuristics. In the area of changing problem representations (2), we have implemented a system, Baggy, for transforming similar objects with different names into bags of objects when these objects are handled identically. We are also in the process of creating a system, Macry, which creates macro-operators that can replace their constituent operators without loss of solution optimality. In the area of heuristics and control knowledge(3), we have implemented both a system, GRIDA*, which uses a greedy algorithm to cheaply find a good combination of heuristics for the current problem and have implemented a system, AGPS, which tries to find complementary sets of heuristics by tuning the parameters it uses to generate and select complementary heuristics. In the area of algorithms(4), we have created an algorithm for doing bidirectional heuristic search which holds the promise to guarantee that the first solution it finds is optimal.

We have broken our research into three phases. The first phase was to develop a system that could select, from a large set of heuristics, a subset that would work well with the given problem. The second phase is to develop a system that could select a good combination of representations and heuristics to solve the current problem. The third phase will be to develop a system that given a problem can select a good combination of algorithms, representations, and heuristics. For each of these phases we will need a meta-level architecture that coordinates the adaptation of the algorithms, representations, and heuristics to the current problem and computational environment. Without such coordination, different combinations of these algorithms, representations, and heuristics cannot be created, evaluated, and assembled together into good problem solvers cheaply enough to be effective. The problem is that the creation, evaluation, and assembling all three components can be quite expensive.

The first phase was completed as a part of AOARD Grant 124018 (2012/03/06 - 2014/03/05). The prototype, RIDA*, from the first phase was entered into the 2014 International Planning Competition (IPC) Sequential Optimal track, where it finished fourth overall (out of 16 competitors), but first among comparable, i.e., unidirectional, problem solvers (out of 12 competitors).

The second phase was completed as part of this AOARD Grant (2015/5/27 – 2016/11/26). Unfortunately, the IPC has not been held since 2014. Thus, we cannot compare MSP against a new set of competitors on a new set of problems. However, we can compare it against RIDA* on the problems from the last IPC. MPS, still unidirectional, currently solves 30 more IPC 2014 problems than RIDA*.

We have started our third phase during this grant with an exploration of bidirectional heuristic search algorithms. We believe we have developed a bidirectional heuristic search algorithm that is a significant improvement over current algorithms, at least with regards to the number of nodes expanded. We are currently in the implementation phase and hope to have results soon and a paper ready for the 2017 Symposium of Combinatorial Search (SoCS) conference.

Ultimate Goals and Importance:

The ultimate goal of this research is to develop problem solvers that are able to solve extremely hard problems. Such systems will need to be able to automatically decompose large problems into smaller ones, use multiple levels of abstractions and multiple perspectives, and explore multiple solution approaches in parallel. We believe that being able to build such systems will potentially assist us in performing such difficult tasks as designing air craft, and solving large scale logistics

problems, etc.

This goal, obviously, will not be achieved soon. Our specific aim, for now, is to build components that can be eventually used in such systems. Two types of components that will be needed by these powerful problem solvers are (1) planning components and (2) meta-level architectures that can reason about how to create a planner to best solve such complex problems.

This project explores how to automatically construct a domain-dependent problem solver from a number of toolboxes that contain different algorithm components, representations, and heuristics. So far, we have looked at ways to efficiently create and select from large numbers of heuristics, e.g., thousands of heuristics, and at ways to efficiently create and select from a number of different representations. This involves being able to predict and evaluate the impact of the different combinations of heuristics and representations on a problem solver's ability to solve a given problem.

Experiments, Results, and Discussion:

In this grant we explored the four areas mentioned in our proposal, namely: architecture, representation, heuristics, and algorithms. We believe we have advanced the field in each of these four areas. We describe our progress in these areas below.

Architecture:

The problem that our architecture research addresses is how to keep the combinatorics of exploring combinations of algorithms, representations, and heuristics from overwhelming any benefit that might result from finding better combinations. This phase of our project just explores how to do this for combinations of representations and heuristics. Our paper "Meta Search Through the Space of Representations and Heuristics on a Problem by Problem Basis"[3] describes our approach. Below we give a brief overview.

Our approach formulates the problem of finding a good combination as a meta-level search problem. Our prototype, MSP, uses a hill-climbing approach to explore a "representation" state space, where the nodes represent different representation combinations and edges represent representation changes. MSP has a set of representation change operators. Some of the operators modify the PDDL description of the problem and some modify the SAS+ description. PDDL and SAS+ are common representations used by planners to represent problems and domains. PDDL is the standard language which humans use to describe problems/domains and SAS+ is a standard problem representation input language used by many planners.

Given a node, MSP uses RIDA* to generate the information that it needs to evaluate the node's quality. RIDA* searches for the best combination of heuristics to use with the node's representation and returns the best heuristic combination, its branching factor, and its average time per node. MSP uses this information to predict how deep this combination would be able to search within a given period of time. This depth is MSP's quality metric. MSP picks the node that it predicts can go the deepest (i.e., highest f-level). In general, if any combination will be able to solve the problem, the one that goes deepest will.

MSP is given an upper limit on how long it has to solve the problem. MSP splits that time limit into (1) how much time it will use to find a good combination of representations and heuristics; and (2) how much time it will use to solve the problem with the selected representations/heuristics. Currently, MSP gives both parts equal time.

There is a tradeoff between how much time RIDA* has to "evaluate" the node's representation and how accurate its prediction of the best heuristic combination. MSP has an estimate of the amount of time RIDA* will need in order to get a useful evaluation. This time is passed to RIDA* along with the description of the problem. RIDA* uses that time to manage its own activities. If RIDA* exceeds that time limit then MSP kills it, and assumes that the representation is too bad to use. RIDA* often finishes evaluating a node before its time limit. In this case, it will immediately return its information. MSP keeps exploring the "representation" space until either it has hit a plateau or it

runs out of time for finding a good combination. If it runs out of time and MSP killed RIDA* for every representation then MSP uses the original representation and a default state-of-the-art heuristic.

When MSP finishes its evaluations of the representations, it selects the node with the best evaluation and uses its representation and set of heuristics to solve the problem. Depending on the nature of the representation changes, at the end the solution may need to be translated back into the original representation.

Representation:

MSP uses a number of representation change operators, some of them were developed by others and are not described in this document. Baggy, Macry and EIT were developed specifically for this grant. Baggy and Macry affect the PDDL-level description, while EIT affects the SAS+ description. These are briefly described below.

PDDL-level Changes:

Transforming Sets of Individually Named Objects into Bags of Anonymous Objects (Baggy):

The complexity of solving a planning problem grows exponentially with the number of distinct objects. For example, if there is an operator with three parameters of object type T and there are four objects of type T, then there are 64, i.e., 4^3, ways to instantiate those parameters. If two more T-type objects are added to the problem then there are 264, i.e., 6^3, ways. Often when there are large numbers of a type of object, their individual identity is not important. For example, when you have a large number of screws in a jar, they do not usually have individual designations, they are usually just categorized into number, size and type (e.g., there are five 2 inch Philips head screws). Transforming a problem where there are there are many individually designated objects into one where they are represented as bags (sets with counts of identical elements) exponentially reduces the size of the search space. Not all object types are transformable into bagged types, they need to satisfy various criteria that ensure that the transformation preserves solution optimality.

When Baggy [2] is given a problem and domain, it analyses the problem and domain to determine whether any of the object types can be transformed into a bagged type. If so, Baggy then translates the problem representation, this involves changing the types, the predicates, and the descriptions of the affected objects and operators and the problem description. The planner uses this new representation of the problem/domain to solve the problem. When the planner finds a solution in this new representation, it must translate that solution back into the original representation before returning it to the user.

Most planners and heuristics find bagged representations difficult to use. Modern planners rely on their translator (from the PDDL user representation to the SAS+ planner representation) being able to find invariants in the search space which allows them to prune away large portions of the space. However, since predicates involving bagged types cannot be handled by translators, we have created the EIT translator to handle bagged predicates. EIT is described below in the "SAS+-level Changes" section.

Many modern heuristics use a technique called "delete-relaxation". Delete relaxation does not handle counts and consequently performs poorly for problems where bags are involved. Fortunately, there are a few heuristics which are not as adversely affected by bagged representations. However, these heuristics tend be worse than the delete-relaxation heuristics. We have made some initial experiments in creating variations of these delete-relaxation heuristics that can better handle these bagged representations.

Macro-Operators (Macry):

One more PDDL-level representation change that we are currently looking into is the automated creation of macro-operators. Our approach takes a problem and domain description and, where appropriate, creates macro-operators which replace their constituent operators while still guaranteeing

optimality. Macry's approach also guarantees that its macro-operators decrease the size of the search space. We are in the middle of implementing this process and in 2018 will write a paper describing it.

SAS+-level Changes:

Enhanced Invariant Translator (EIT):

As mentioned above, modern planners use translators to translate the user-given PDDL problem and domain descriptions into the internal representation used by the planner. For many planners, and for the Fast Downward planner that we use, SAS+ is the input representation language used to pass information from the preprocessor to the planner. One function of the translator is to discover as many invariants as possible to prune the search space. However, the cost of finding invariants grows exponentially with respect to the number of arguments used by the domain's predicates. To keep the cost of finding invariants practical, the translator limits the number of arguments in the predicates that it processes when looking for invariants. Baggy's transformation of the predicates increases their number of arguments beyond this limit. Because Baggy produced the bagged types, it knows many of their invariants and passes them directly to the translator. Thus enabling the planner to use them to prune the search space. These invariants are passed to EIT which uses them in its translation to SAS+.

Heuristics:

In the 2014 IPC, the generation of RIDA*'s Pattern Database heuristics (PDBs) took approximately 75% of the total time RIDA* used to solve a problem (only approximately 2% of the RIDA*'s total time was used in solving the problem, the remaining 23% was used to evaluate the different heuristic combinations). The large cost of generating the PDBs limits how many meta-level nodes can be generated and evaluated by MSP in its time limit. We have explored different ways to lower the cost of generating the PDBs.

In particular, we explored two different approaches. One approach, GRIDA*[1], is to do a greedy search for a good set of heuristics for the given problem and the other, AGS[4], is to more intelligently create a smaller number of heuristics that are more tailored to the given problem and which work well together. A brief description is given below.

The greedy approach trades the quality of the combination of heuristics, used by the planner, for lowering the cost of finding that combination. This approach is described in more detail in [1]. Our original 2014 RIDA*prototype searched for a globally “optimal” combination of heuristics, which had a time limit in which to find a combination. It ordered the search by the number of heuristics in the combination, starting with combinations with a single heuristic and working upwards with increasing numbers of heuristics. With this approach RIDA* would usually find combinations of 2 or 3 heuristics (up to a max of 10 heuristics) usually from a set of 45 heuristics. With the greedy approach, GRIDA*often found combinations of 40 or more heuristics, from sets of thousands of heuristics.

Even being able to select from thousands of generated heuristics, GRIDA* was not able to do much better than RIDA*. In benchmark experiments, GRIDA*solved 219 problems and RIDA*solved 210. Our second approach, AGPS, attempts to create better combinations of heuristics. This research is described in [4] which was submitted to the 2017 IJCAI. The main idea was to dynamically adapt our procedure to generate heuristics, based on feedback about how complementary our heuristics were and how frequently we were finding new complementary heuristics. This approach seems to be superior to existing approaches, solving 37 more problems than SYMBA*, the winner of the 2014 International Planning Competition (IPC).

Algorithms:

Algorithms were our final problem solver component (representations and heuristics being our other problem solver components). We have just begun research in this area, having concentrated

primarily on developing techniques to transform representations of problems to make them easier for our problem solver to solve.

The types of algorithms we have focused on are bidirectional heuristic search algorithms. Bidirectional blind search exponentially reduces the size of a problem's search space, from b^d to $b^{d/2}$, where b is the branching factor and d is the length of the optimal solution path. Heuristic search also exponentially reduces the size of the search space. If the average value of a heuristic, h, for a randomly chosen state is n then using the heuristic will reduce the size of the search space from b^d to b^{d-n}. Ideally, using a heuristic while doing bidirectional search would reduce it to $b^{(d-n)/2}$. Only recently have bidirectional heuristic search algorithms come anywhere close to this. Until the last year or two, these algorithms were normally beaten by either unidirectional heuristic search (e.g., A*), bidirectional blind search, or by both.

We are currently working in this area and believe we have found a bidirectional heuristic search algorithm that guarantees that the first solution found is guaranteed to be optimal. To the best of our knowledge this is the first such algorithm with this capability. We will be submitting this work[5] for publication at the Symposium on Combinatorial Search (SoCS) conference this year.

List of Publications and Significant Collaborations that resulted from your AOARD supported project:

b) **Papers published in peer-reviewed conference proceedings:**

[1] Lelis, Levi, Santiago Franco, Marvin Abisrror, Mike Barley, Sandra Zilles, and Robert Holte. "Heuristic Subset Selection in Classical Planning." In *Proc. IJCAI*, pp. 3185-3191. 2016.

[2] Riddle, Pat, Jordan Douglas, Mike Barley, and Santiago Franco. "Improving Performance by Reformulating PDDL into a Bagged Representation." In *Heuristic Search in Domain Independent Planning (HSDIP*), pp. 28-36. 2016.

e) **Manuscripts submitted but not yet published:**

[3] Fuentetaja, Raquel, Mike Barley, Daniel Borrajo, Jordan Douglas, Santiago Franco, and Patricia Riddle. "Meta Search Through the Space of Representations and Heuristics on a Problem by Problem Basis." Submitted to *IJCAI,* 2017.

[4]Franco, Santiago, Alvaro Torralba, Levi Lelis, and Mike Barley. "On Generating Complementary Pattern Databases." Submitted to *IJCAI*, 2017.

[5]]Barley, Mike, Patricia Riddle, and Carlos Linares. "Optimal First-Collision Bidirectional Heuristic Search." To be submitted to *Symposium on Combinatorial Search (SoCS)*, 2017.

f) **Significant collaborations resulting from this work**: Two significant collaborations have resulted from this work. One is with Daniel Borrajo and Raquel Fuentetaja, Universidad Carlos III de Madrid on the meta-level search architecture for finding good combinations of representations and heuristics on a problem-by-problem basis. The other is with Carlos Linares also from Universidad Carlos III de Madrid on developing effective bidirectional heuristic search algorithms. Both of these collaborations are continuing on a on-going basis for the foreseeable future.

Attachments: Publications a), b) and c) listed above if possible.